J. Symbolic Computation (1989) 7, 487–507

A Deductive Database Based on Aristotelian Logic*

EYAL MOZES

Department of Computer Science, Stanford University, Stanford, CA 94305, *USA*

(*Received* 20 *September* 1987)

The paper proposes a new type of deductive database, whose structure is based on Aristotle's rules of the syllogism. This type of database demonstrates the advantages of Aristotelian logic over more modern formalisms for applications in which natural and informative interaction with human users is important. The structure and deductive procedures of the database are described and compared to other work in the area, and the unique capabilities of its user interaction are demonstrated by examples from a session with a prototype system.

1. Introduction

Aristotelian logic was created for the purpose of understanding and practically guiding actual human thought; this was not the purpose of modern, mathematical logic. As a result, Aristotelian logic has some advantages over other systems of logic for applications in which natural and informative interaction with human users is important. The proposed database is one such application, demonstrating these advantages.

The database holds information about a set of objects and their properties. Interaction with the user includes the following abilities, which are unique to this database and are not possible within the framework of other deductive databases:

(1) The ability to give natural-language explanations of the deductions, constructed for each case in a form that fits the deduction and looks natural and convincing to the user.
(2) The ability to volunteer information, in answer to yes/no questions, if a stronger or weaker version of the "yes" answer can be proved.
(3) The ability to point out results that can't be proved but seem to be likely possibilities.
(4) The ability, in answer to yes/no questions, to suggest "missing rules", i.e. rules that, if added to the database, will allow proving a "yes" answer.
(5) The ability to suggest instances in which non-deductive forms of reasoning, such as analogy or induction, are likely to be useful.

2. Structure of the Database

The database consists of a set of constants, representing objects, and a set of relations (there are no functions). Information about them is expressed by specific facts, consisting of relations applied to constants, and by deductive rules.

* This paper describes the author's Master's thesis, written at the Weizmann Institute of Science, under the supervision of Professor Amir Pnueli (Mozes, 1987).

0747–7171/89/050487+21 $03.00/0

A unary relation can represent either a concept to which objects can belong (for example, "man" or "animal") or a single possible attribute for an object (for example, "mortal" or "wise"). Relations with higher arities represent attributes with one or more measurement values (for example, "length" and "age" can both be represented as binary relations)†. The database contains information indicating what each relation represents, and also indicating some English grammatical properties of the constants and relations, to be used in constructing the natural-language explanations.

A rule consists of: a "subject", which is a conjunction of one or more (perhaps negated) relations applied to variables and constants; a "predicate", which is a single (perhaps negated) relation applied to variables and constants; and a rule type, which indicates the type of connection between the subject and the predicate (a more general database can allow the predicate to be a disjunction of several relations). When writing the rule, the predicate is written first, then the rule type, and then the subject.

There are four possible rule types, corresponding to Aristotle's four types of propositions: type A, which asserts that all objects fulfilling the subject also fulfill the predicate; type E, which asserts that none of the objects fulfilling the subject fulfill the predicate; type I, which asserts that some of the objects fulfilling the subject also fulfill the predicate; and type O, which asserts that some of the objects fulfilling the subject do not fulfill the predicate. For example: "mortal(X) A man(X)" means that every man is mortal; "man(X) E dog(X)" means that no dog is a man; "mammal(X) I animal(X)" means that some animal is a mammal; "baby(X) A (man(X) $\wedge$ age(X,1))" means that every man with age 1 is a baby.

Every rule is about *something*, i.e. all rule types imply that there exist some objects fulfilling the subject.

There are two types of queries to the database: yes/no questions, which state a specific fact or a deductive rule and ask whether it can be proven true or false; and retrieval requests, which give a conjunction of one or more relations applied to variables and constants, and ask for values of the variables that fulfill these relations.

After getting an answer to a query, the user may ask for an explanation of the deduction. If the query was a retrieval request, the user may specify a constant and ask why this constant was or was not retrieved. If the query was a yes/no question, and it couldn't be proven either true or false, the user may also ask for "missing rules".

Negative information can be expressed in the database explicitly, through rules of type E and O, and also through negative specific facts (i.e. asserting that a certain relation does not apply to a certain tuple). Relations in a rule, and in a query, can also be negated. Negation is expressed by the $\neg$ character.

In addition to proving negative knowledge explicitly, the database also uses a generalised form of negation as failure; negation of a fact requires both failure to prove it and failure to point it out as a likely possibility. The user is informed when negation as failure was used in proving a result. Thus, something is false if it can be explicitly proven false or if there is no evidence—not even inconclusive evidence—that it is true; and the user knows which one it is in each case.

Incomplete information can be expressed in the database by null values. For example, "length(**a**,null)" means that object **a** has some length, but its length is not known. This

† The measurement values are often numerical, but not always. For example, "shape" can be a binary relation in which the measurement values are "triangular", "circular", etc.

In a more general database, there will also be relations representing relationships between objects (for example, "father" or "bigger" can be binary relations). See section 6.1 for a discussion of how such relations are treated in Aristotelian logic.

meaning of a null value generalises naturally to all contexts in which a relation can appear; for example, "¬ length(**a**,null)" means that object **a** does not have a length; or if "length(X,null)" appears in the subject of a rule (where X is a variable), the rule refers to all (or, if it is of type I or O, some) objects that have a length, regardless of what the length is. When the value of an attribute is numerical, the database allows a more general form of incomplete information: giving a range as the value of an attribute; for example, "length(**a**,10–30)" means that **a** has a length between 10 and 30. Again, the meaning of a range as a value generalises naturally to all contexts in which a relation can appear.

3. Deductive Procedures

The deductive procedures of the database are based on the theory of the syllogism, as developed by Aristotle (Ross, 1949) and presented in many logic textbooks, for example by Joseph (1916) and by Maritain (1937). Since one cannot possibly do justice to this theory in the following brief discussion, the reader is urged to study one of these textbooks.

I start with explanations of some basic terms of Aristotelian logic:

(1) Quantity—the "quantity" of a rule means whether it gives information about every object fulfilling the subject or only some of them. Types A and E have "universal quantity", I and O have "particular quantity".
(2) Quality—the "quality" of a rule means whether it is affirmative or negative. Types A and I are affirmative, E and O are negative.
(3) Term—the subject or predicate of a rule is a "term".
(4) Distribution—a term in a rule is "distributed" if the rule gives information about all objects belonging to this term. The subject is distributed in rules with universal quantity, undistributed in rules with particular quantity; the predicate is distributed in negative rules, undistributed in affirmative rules.

3.1. IMMEDIATE INFERENCES

There are several procedures of "immediate inference", i.e. using one rule for deriving another rule.

(1) Obversion—it is possible to reverse the quality of a rule and negate its predicate (or, if it was negated, make it positive). For example, the rule "mortal(X) A man(X)" (every man is mortal) can be obverted into "¬ mortal(X) E man(X)" (no man is immortal).
(2) Conversion—it is sometimes possible to exchange the subject and predicate in a rule. This is possible in rules of type E and I; for example, the rule "mammal(X) I animal(X)" can be converted into "animal(X) I mammal(X)". A rule of type A can be "converted accidentally", by turning it into an I proposition and then converting.
(3) Contraposition—it is sometimes possible to exchange the subject and predicate in a rule and negate both of them. This is possible in rules of type A and O; for example, the rule "mortal(X) A man(X)" can be contraposed into "¬ man(X) A ¬ mortal(X)". Contraposition is equivalent to obversion, conversion and another obversion.
(4) If the subject in a rule of type I is a conjunction of several relations, the rule can be "partially converted" by exchanging one of these relations with the predicate.

It is also sometimes possible, from the truth or falsity of one rule, to deduce the truth or falsity of another rule with the same terms and a different type, as follows:

(1) Types A and E are "contraries", i.e. they can both be false but they can't both be true.
(2) Types I and O are "sub-contraries", i.e. they can both be true but they can't both be false.
(3) Types A and O are "contradictories", i.e. always exactly one of them is true and the other false. Types E and I are also contradictories.
(4) Type I is a "subaltern" of type A, i.e. it is impossible for A to be true and I to be false. Also, type O is a subaltern of type E.

3.1.1. Widening or narrowing a term

There are three cases in which we say that one term is "wider" than another term.

(1) A conjunction of several relations is narrower than a term that contains only one, or only part, of these relations.
(2) A term in which one value of a non-negated (negated) relation is a numerical range, is wider (narrower) than a similar term in which there is in the same place a narrower range or a constant from that range.
(3) A term in which one value of a non-negated (negated) relation is a null value, is wider (narrower) than a similar term in which there is in the same place a constant or a numerical range.

A combination of several such differences, in the same direction, is also possible.

Any undistributed term can be replaced by a wider term. For example, from the rule "idiot(X) I (man(X) ∧ age(X,30–50))" (some man with age between 30 and 50 is an idiot) we can deduce the rule "idiot(X) I (man(X) ∧ age(X,20–60))", or the rule "idiot(X) I man(X)".

Any distributed term can be replaced by a narrower term, if we know that there exist some objects fulfilling the narrower term. For example, from the rule "mortal(X) A man(X)" we can deduce the rule "mortal(X) A (man(X) ∧ age(X,30–50))".

3.2. SYLLOGISMS

A syllogism is a deduction of a new rule from two previously known rules.

For purposes of the syllogism, a specific fact can be regarded as a rule in which the constant values are the subject. For example, "Man(Socrates)" can be regarded as a rule of type A in which the subject is "Socrates(X)" and the predicate is "Man(X)". In the same way, every positive fact can be regarded as a rule of type A, and every negative fact as a rule of type E.

There are three sorts of restrictions a syllogism must fulfill to be valid—"terminological restrictions", dealing with the terms appearing in the syllogism; "qualitative restrictions", dealing with the quality of the premises and the conclusion; and "quantitative restrictions", dealing with the quantity of the premises and the conclusion and with the distribution of the terms.

3.2.1. Terminological restrictions

There must be exactly three terms in the premises and the conclusion, each one appearing twice. The subject of the conclusion is the "minor term"; the predicate of the

conclusion is the "major term"; and the third term, which appears only in the premises, is the "middle term". The premise in which the minor term appears is the "minor premise", and the premise in which the major term appears is the "major premise".

If the subject of the major premise is a conjunction of several relations, and some of them are predicates of true type A rules in which the minor term is the subject, then they can be ignored for the purposes of this syllogism. For example, if the minor premise is "age(Joe,1)", the major premise is "baby(X) A (man(X) ∧ age(X,0–2))", and "man(Joe)" is true, then the relation "man(X)" in the major premise can be ignored, i.e. the major premise can be treated in this syllogism as if it was "baby(X) A age(X,0–2)" (and from this, as described in section 3.1.1, we can deduce "baby(X) A age(X,1)"; this creates a valid syllogism with the conclusion "baby(Joe)").

3.2.2. Qualitative restrictions

(1) At least one premise must be affirmative.
(2) If both premises are affirmative, the conclusion should also be affirmative.
(3) If one premise is negative, the conclusion should also be negative.

3.2.3. Quantitative restrictions

(1) Any term that is undistributed in the premises must also be undistributed in the conclusion.
(2) The middle term must be distributed in at least one premise.
(3) At least one premise must be universal.
(4) If one premise is particular, the conclusion should also be particular.

Quantitative restrictions 3 and 4 are not independent, but can be deduced from the others.

3.3. CAPABILITIES OF THE USER INTERACTION

Use of the above procedures allows the following unique capabilities in interaction with the user:

3.3.1. Natural-language explanations

Aristotle made two main classifications of the various syllogisms:

(1) Moods. The mood is determined by the types of the two premises. If both premises are of type A, the syllogism is of mood AA; if the major premise is of type E and the minor premise of type I, the syllogism is of mood EI; etc.
(2) Figures. The figure is determined by the place of the middle term in the premises. If the middle term is predicate of the minor premise and subject of the major premise, the syllogism is of the first figure; if the middle term is predicate of both premises, the syllogism is of the second figure; if the middle term is subject of both premises, the syllogism is of the third figure.†

    According to Joseph's analysis (and contrary to Aristotle's own views), each of the figures represents a different type of reasoning: the first figure represents direct reasoning, based on the principle that whatever satisfies the condition of a true rule

† There is some controversy among logicians about the existence of a fourth figure, but it isn't important for our purposes; see Joseph (1916) or Maritain (1937) for details.

falls under the rule; the second figure represents indirect reasoning, by reductio ad absurdum; the third figure represents demonstrating existence by an example.

Using these two classifications, it is easy to construct, for each deductive step, a natural-language sentence explaining it that looks natural and convincing to a human user; these sentences are used to give a full natural-language explanation of the deduction. For example, a syllogism of mood AA, first figure will be explained by a sentence of the form "Because S is M, it is P"; a syllogism of mood EA, second figure will be explained by a sentence of the form "S can't be P, because then it wouldn't be M"; a syllogism of mood AA, third figure will be explained by a sentence of the form "Some S is P; for example M"; etc. The form is filled in by each of the three terms, according to its type (constant or relation, concept or attribute) and its grammatical properties (such as its plural form and the pronoun that can stand for it). See section 5 for examples of such sentences.

Note that the four types of rules allow representing each piece of knowledge in its most natural form, and this helps in constructing good explanations. For example, suppose you want to represent the information that only a man can be responsible†. In most deductive databases, or in the PROLOG language, this would be represented by the rule "man(X):—responsible(X)"; similarly, it is possible to represent this knowledge in the proposed database by the rule "man(X) A responsible(X)". But this is clearly an unnatural form for this rule. The natural expression of this information in English is *not* "every responsible being is a man", but "only a man can be responsible"; and there is a reason for this—its natural use is not to conclude that someone is a man because you know that he is responsible, but to conclude that something (for example, your cat) is not responsible because you know that it's not a man. The proposed database gives us a natural way to express this rule: "responsible(X) E ¬man(X)". This form is equivalent to the previous one in the results you can deduce from it, but it will produce different, and better, natural-language explanations for the deductions.

3.3.2. Stronger or weaker results

It is possible that a user, in a yes/no question, asked about a rule with universal quantity (i.e. of type A or E), and the database, while unable to prove it, was able to prove a corresponding rule with particular quantity. It is also possible that the user asked about a rule with particular quantity and the database was not only able to prove it but also able to prove a corresponding rule with universal quantity. In both cases, this information can be volunteered to the user.

3.3.3. Pointing out likely possibilities

According to the list of restrictions on valid syllogisms, invalid syllogisms can be classified by the failures that they commit, i.e. by the restrictions that they violate.

In particular, let us look at syllogisms that violate quantitative restriction 2 (and maybe also 3 or 4). Such syllogisms commit the "fallacy of the undistributed middle", i.e. the middle term is undistributed in both premises. This is a fallacy because, since each premise refers to only some of the objects belonging to the middle term, we cannot be sure that they refer to the *same* objects, and, therefore, we are not sure that the middle term performs its job of linking the two terms of the conclusion.

† The word "man", of course, is used here in the generic meaning, which includes both sexes.

However, while we cannot be *sure* that both premises refer to the same part of the middle term, it is *possible* that they do. This means that, if a conclusion was reached by committing the fallacy of the undistributed middle once, and no other fallacy, and the undistributed middle term is not negated (i.e., it refers to the objects that belong to some specified concept or have some specified attributes), then we do have some, inconclusive, evidence for the truth of the conclusion; in that case, the conclusion should not be presented as proven, but can be pointed out as a likely possibility.

3.3.4. "Missing rules"

The database can use a chain of syllogisms of any length in reaching a conclusion. In trying the possibilities for such chains, some paths are found not to lead to a valid proof. The restrictions on valid syllogisms can be used, at the end of such a chain, to find whether some additional rule, consistent with the other rules in the database, would, if added to the database, allow creating a valid proof†. These "missing rules" can then be reported to the user.

Reporting of such a "missing rule" can be useful if the user was assuming this rule to be true, or knew it to be true, but it was not specified in the database. It is also useful if the "missing rule" is a stronger version of a rule known to be true, or if there exists some inconclusive evidence that it is true.

3.3.5. Suggesting instances for non-deductive reasoning

As explained above, a result can be suggested as a likely possibility if it is the conclusion of a syllogism committing the fallacy of the undistributed middle. Sometimes, that invalid syllogism is in the first figure, and its major premise is itself the conclusion of a valid syllogism in the third figure (see, for example, the suggestion of Fido as a possibility in example 1 in section 5). In such a case, this is a simple form of reasoning by analogy.

Also, when a conclusion is reached by a valid syllogism in the third figure, that conclusion always has particular quantity, and the syllogism suggests the possibility of establishing a stronger version (i.e. the corresponding rule with universal quantity) by induction‡.

4. Comparison to other work

4.1. OTHER WORK ON DEDUCTIVE DATABASES

The general structure of the database is similar to that described by Gallaire *et al.* (1978; 1984); one difference is that, instead of only one type of rule (implication), the database allows four types (of these, A is the closest in its meaning to implication, but is not exactly the same since it implies that there exist some objects fulfilling the subject§).

† Such incomplete paths, handled in this way, are known in Aristotelian logic as "enthymemes".

‡ Such induction is made more plausible when we can find more such syllogisms with the same conclusion but with different middle terms. It is the task of inductive logic, and outside the scope of the database, to determine when we can be certain that the induction is justified.

§ This implication is consistent with the way people naturally think; today, people with a background in modern formal logic reject it theoretically, but still use it in everyday reasoning. See Veatch (1952, pp. 243–263), Maritain (1937, pp. 225–233), or the full thesis (Mozes, 1987, pp. 22–23) for more details on the rationale of this implication.

The structure is similar to what Gallaire *et al.* (1984) refer to as "definite deductive databases", i.e. the predicate must be a single relation.

The treatment of negative information is also different. The databases described by Gallaire *et al.*, as well as conventional relational databases and logic-programming languages such as PROLOG, use only negation as failure. In contrast, the proposed database allows expressing and proving negative knowledge *explicitly*, avoiding non-monotonicity and the need for assuming complete knowledge; and it also uses a generalised form of negation as failure.

In its scope and the type of information it contains, the proposed database is similar to Lipski and Marek's "information storage and retrieval systems" (Lipski & Marek, 1977; Lipski, 1979). One difference is that, in Lipski and Marek's system, each attribute must be defined over all objects. In the proposed system, this assumption is not made; whether an attribute is defined over an object is itself a piece of information that the database may contain (for unary relations, this is the *only* information; for higher arities, this information is expressed by null values). The allowed type of incomplete information is a slightly restricted version of that allowed by Lipski.

However, the most important difference is in the motivation. Most previous work on deductive databases concentrated on technical issues of completeness and expressive power, or on issues of efficiency. In contrast, the main motivation of this paper is improving the interaction with the user.

4.2. OTHER WORK ON ARISTOTELIAN LOGIC

Aristotelian logic was used only once before in any application in computer science—in KONSDED, an inference engine used for consistency-checking and deduction of new relationships in the knowledge-base of the medical expert system CADIAG-1 (Barachini, 1984; F. Barachini, personal communication, 1985; K.-P. Adlassnig, personal communication, 1985). However, this use of Aristotelian logic is in a part of the system that does not interact with the user (in the inference engine, rather than in the explanation facility where it could have been of value), and it therefore does not demonstrate the strong points of Aristotelian logic; as a result, it had no special advantage over more modern formalisms, and later versions of KONSDED were modified to use predicate calculus (Adlassnig *et al.*, 1985).

Aristotelian logic is also mentioned in a recent paper about the binary set containment inference problem (Atzeni & Parker, 1986). The authors see the possibility of using Aristotelian logic for the binary set containment inference problem (which uses propositions of the four types, only with unary relations, with only one relation in the subject, and with no constants), but, again, don't see any special advantage in it over predicate calculus, or its use for interaction with human users.

5. Examples

A prototype of the proposed database was developed and tested successfully, and has demonstrated all the unique capabilities claimed above. Following are some examples from a session with this prototype system.

In the following examples, text in curly braces explains the meaning of rules and queries, and is not part of the session.

The contents of the database during the session were:

SPECIFIC FACTS:

man(Socrates)
wise(Socrates)
man(Joe)
age(Joe,1)
man(John)
age(John,15)
man(Peter)
age(Peter,40)
dog(Fido)
work_of_art(Mona_Lisa)

RULES:

predicate	*rule-type*	*subject*	
animal(X)	A	man(X)	{Every man is an animal}
animal(X)	A	dog(X)	{Every dog is an animal}
mortal(X)	A	man(X)	{Every man is mortal}
mortal(X)	E	work of art(X)	{No work of art is mortal}
man(X)	E	dog(X)	{No dog is a man}
responsible(X)	E	¬man(X)	{No non-man is responsible}
responsible(X)	A	wise(X)	{Everyone who is wise is responsible}
responsible(X)	E	baby(X)	{No baby is responsible}
responsible(X)	I	adult(X)	{Some adult is responsible}
responsible(X)	E	idiot(X)	{No idiot is responsible}
age(X,0–120)	A	man(X)	{Every man has age 0–120}
baby(X)	A	(man(X) ∧ age(X,0–2))	{Every man with 0–2 is a baby}
adult(X)	A	(man(X) ∧ age(X,21–120))	{Every man with age 21–120 is an adult}
idiot(X)	I	(man(X) ∧ age(X,30–50))	{Some man with age 30–50 is an idiot}

EXAMPLE 1

> mortal(X) ∧ ¬responsible(X) {find an object which is mortal and not responsible}

Joe
By negation as failure: John
Possibility: Fido

> Explain Joe

Because Joe is a man, he is mortal.

Because Joe is a man with age 1, he is a baby.
Because Joe is a baby, he is not responsible.

> Explain Fido

Because Fido is a dog, it is an animal.
Some animal is mortal; for example, the men.
Because Fido is an animal, it is, perhaps, mortal.

Because Fido is a dog, it is not a man.
Because Fido is not a man, it is not responsible.

> Explain Socrates {why wasn't Socrates retrieved?}

Because Socrates is wise, he is responsible.

> Explain Peter {why wasn't Peter retrieved?}

Because Peter is a man with age 40, he is an adult.
Because Peter is an adult, he is, perhaps, responsible.

EXAMPLE 2

> ¬man(X) {Find an object which is not a man}

Fido, Mona_Lisa

> Explain Fido

Because Fido is a dog, it is not a man.

> Explain Mona_Lisa

Because Mona Lisa is a work of art, it is not mortal.
Mona Lisa can't be a man, because then it would be mortal.

EXAMPLE 3

> age(Socrates,?) {What is Socrates's age?}

Range 2–120

> Explain

Because Socrates is a man, he has age in range 0–120.

Because Socrates is wise, he is responsible.
Socrates can't be a baby, because then he wouldn't be responsible.
Socrates can't have age in range 0–2, because then he would be a baby.

EXAMPLE 4

> mortal(X) A animal(X) {Is every animal mortal?}

Unknown. But it is known that some animal is mortal.
It may be possible to establish by induction that every animal is mortal.

> Explain

Some animal is mortal; for example, the men.

> Missing

No possible addition of a rule can establish your conclusion.

EXAMPLE 5

> man(X) A animal(X) {Is every animal a man?}

No.

> Explain

Some animal is not a man; for example, the dogs.

EXAMPLE 6

> wise(X) A (man(X) ∧ age(X,30–90)) {Is every man with age between 30 and 90 wise?}

No.

> Explain

We know that some man with age in range 30–50 is an idiot.
Because he is an idiot, he is not responsible.
He can't be wise, because then he would be responsible.

EXAMPLE 7

> work_of_art(X) E animal(X) {Is no animal a work of art?}

Unknown. But it is known that some animal is not a work of art.

> Explain

Some animal is mortal; for example, the men.
It can't be a work of art, because then it wouldn't be mortal.

> Missing

Can establish your conclusion if you add the rule:
mortal(X) A animal(X)

6. Applicability of Aristotelian logic

To my knowledge, the best detailed presentation of the philosophical base of Aristotelian logic, and of the basic reasons for its advantages over modern, mathematical logic, is by Veatch (1952). The full thesis (Mozes, 1987) contains a discussion of the philosophical issues, based on Veatch's book.

6.1. POSSIBLE EXTENSION: RELATIONS REPRESENTING RELATIONSHIPS BETWEEN OBJECTS

The common view about relations representing relationships between objects, e.g. "father" or "bigger", is that facts and inferences involving such relations are a different form of reasoning than the classical subject-predicate propositions and syllogisms. Some mathematical logicians regard the syllogism as reasoning with unary relations; others regard it as a special case of reasoning with transitive relations (i.e., a syllogism of the form "every a is b; every b is c; therefore every a is c" is regarded as isomorphic to arguments like "a is bigger than b; b is bigger than c; therefore a is bigger than c"); but all agree that the syllogism is only one special case of deductive reasoning. The database described in this paper would therefore be criticised as limited to only one form of deduction.

However, an examination of the way relationships between objects are treated in actual human thought would show that there are no special problems in extending this database to deal with such relations.†

† The following is based on Veatch's (1952) discussion, particularly pp. 344–358; the main points were already recognised by Aristotle, and are also mentioned by Joseph (1916, pp. 341–342).

Suppose that, in a more general version of our database, we have the relation "bigger", and want to express the fact that it is transitive; this would be expressed as "bigger(X,Y) A (bigger(X,Z) ∧ bigger (Z,Y))". If we also have the facts "bigger(**a**,**b**)" and "bigger(**b**,**c**)", we can use them to conclude "bigger(**a**,**c**)". Is this a different form of reasoning than the syllogism?

"Bigger(X,Y) A (bigger(X,Z) ∧ bigger(Z,Y))" is a rule about relational complexes, i.e. about objects related to one another in certain ways; it states that any relational complex consisting of three objects such that X is bigger than Z and Z is bigger than Y, will also be such that X is bigger than Y. This rule was used as a premise, together with the other premises that told us the relational complex of **a**, **c** and **b** fulfills the condition, to conclude that it also has the property that **a** is bigger than **c**.

This means that the inference is a syllogism; it has *exactly* the same form as concluding "mortal(Socrates)" from "mortal(X) A man(X)" and "man(Socrates)", and the only difference is that the minor term is a relational complex rather than a single object. The same is true of any such inference, including inferences with relations of arities larger than 2.

As described in section 3.2, specific facts are regarded, for purposes of the syllogism, as rules in which the constant values are the subject. Note that exactly the same can be done in reasoning about relational complexes. For example, the fact "bigger(**a**,**b**)" can be regarded as a rule of type A in which the subject is the tuple "**a**,**b**" and the predicate is "bigger".

We see, therefore, that the rules of the syllogism, as used in the database described in this paper, are an appropriate general model for deductive inference. A more general database, which would allow knowledge and inference about relational complexes, will, of course, be able to express a wider range of facts, and may present some interesting problems in its implementation, but it will not add any new theoretical aspects or problems relevant to logic.

6.2. AREAS OF APPLICABILITY

The advantages of Aristotelian logic demonstrated above are based on its correspondence to actual human thought. This suggests two broad areas to which Aristotelian logic can be applied:

(1) Applications in which interaction with human users is important.
(2) AI applications, which are concerned with simulating human thought, and in which some of the capabilities described above, such as suggestion of likely possibilities and connection to induction, are important.

This paper is concerned mainly with area 1. Aristotelian logic has so far received no attention from computer scientists in these areas (as I mention above, the only previous works in computer science using Aristotelian logic are KONSDED and the binary set containment inference problem, which do not belong to either of the two areas).

A third area, on which many applications of logic in computer science have so far concentrated, is mathematical theorem-proving. In this area, the special capabilities made possible by Aristotelian logic are not important. Some features of theorem-proving systems, such as functions, would probably be hard to fit into the framework of Aristotelian logic. The procedures presented in section 3 are more complex than those of other logic programming systems, and any implementation of them will probably be less

efficient; also, the treatment of explicit negation in these procedures is not complete (see example in section 7.2). For these reasons, Aristotelian logic will probably not be useful for mathematical theorem-proving systems.

7. Aristotelian Logic as an Extension of Other Deductive Databases

The deductive procedures of this database, which are based on Aristotelian logic, as described in section 3, are very different from those of Gallaire *et al.* (1984), or of logic-programming languages such as PROLOG, and it may not be clear how they can be compared. Two questions that will make this comparison clearer are:

(1) Can we create a system of rules for deduction that will be an extension of logic programming and will provide the same deductive powers as the Aristotelian-logic database?
(2) Can we give a formal semantics for the database that will look like an extension of the "model-theoretic" semantics of other deductive databases or of PROLOG?

7.1. A SYSTEM OF RULES FOR VALID INFERENCE

Several logicians have created "axiomatisations" of Aristotelian logic, i.e. reduced it to a compact set of rules for deduction. The first, and best-known, axiomatisation is by Lukasiewicz (1951); more recently, Atzeni & Parker (1986) also gave such an axiomatisation. All these axiomatisations dealt with a special case: only one relation in the subject, and no constants.

Following is a similar axiomatisation, a system of eleven rules, which also deals with the case of several relations in the subject and with constants. It is an extension of logic programming in two (orthogonal) directions—explicit negation, and particular quantity—and is equivalent, for the purposes of valid inferences, to the procedures described in section 3.

The system is given in terms of types A and I; we can get the equivalent of types E and O by negating the predicate. The $\neg$ character is used for negation and for cancellation of a negation (i.e., $\neg\neg P$ is the same as P).

The system deals only with unary relations. As discussed in section 6.1, the extension to higher arities does not add significant new aspects or problems; it can be done by: a. using a tuple of variables in the place of one variable, and b. using a higher-arity relation with specified values in one or more of the places as a lower-arity relation (see the example below of the relation "age_0_to_2").

The rules are:

(1) If $P, M_1, \ldots, M_j$ are relations or their negations, and s is a constant, then from

$$\begin{array}{c} P(X)\ A\ (M_1(X) \wedge \ldots \wedge M_j(X)) \\ M_1(s) \\ \vdots \\ M_j(s) \end{array}$$

we can deduce

$$P(s)$$

(2) If $P,M_1,\ldots,M_j,S_1,\ldots,S_k$ are relations or their negations, then from

$$P(X) \text{ A } (M_1(X) \wedge \ldots \wedge M_j(X))$$
$$M_1(X) \text{ A } (S_1(X) \wedge \ldots \wedge S_k(X))$$
$$\vdots$$
$$M_j(X) \text{ A } (S_1(X) \wedge \ldots \wedge S_k(X))$$

we can deduce

$$P(X) \text{ A } (S_1(X) \wedge \ldots \wedge S_k(X))$$

(3) If P and S are relations or their negations, then from

$$P(X) \text{ A } S(X)$$

we can deduce

$$\neg S(X) \text{ A } \neg P(X)$$

(4) If $M,P_1,\ldots,P_j$ are relations or their negations, s is a constant, and $1 \leqslant k \leqslant j$, then from

$$M(X) \text{ A } (P_1(X) \wedge \ldots \wedge P_j(X))$$
$$\neg M(s)$$
$$P_1(s)$$
$$\vdots$$
$$P_{k-1}(s)$$
$$P_{k+1}(s)$$
$$\vdots$$
$$P_j(s)$$

we can deduce

$$\neg P_k(s)$$

(5) If $M,P_1,\ldots,P_j,S_1,\ldots,S_l$ are relations or their negations, and $1 \leqslant k \leqslant j$, then from

$$M(X) \text{ A } (P_1(X) \wedge \ldots \wedge P_j(X))$$
$$\neg M(X) \text{ A } (S_1(X) \wedge \ldots \wedge S_l(X))$$
$$P_1(X) \text{ A } (S_1(X) \wedge \ldots \wedge S_l(X))$$
$$\vdots$$
$$P_{k-1}(X) \text{ A } (S_1(X) \wedge \ldots \wedge S_l(X))$$
$$P_{k+1}(X) \text{ A } (S_1(X) \wedge \ldots \wedge S_l(X))$$
$$\vdots$$
$$P_j(X) \text{ A } (S_1(X) \wedge \ldots \wedge S_l(X))$$

we can deduce

$$\neg P_k(X) \text{ A } (S_1(X) \wedge \ldots \wedge S_l(X))$$

(6) If $P,S_1,\ldots,S_j$ are relations or their negations, then from

$$P(X) \text{ A } (S_1(X) \wedge \ldots \wedge S_j(X))$$

we can deduce

$$P(X) \text{ I } (S_1(X) \wedge \ldots \wedge S_j(X))$$

(7) If $P,S_1,\ldots,S_j$ are relations or their negations, and $1 \leqslant k \leqslant j$, then from

$$P(X)\ I\ (S_1(X) \wedge \ldots \wedge S_j(X))$$

we can deduce

$$S_k(X)\ I\ (P(X) \wedge S_1(X) \wedge \ldots \wedge S_{k-1}(X) \wedge S_{k+1}(X) \wedge \ldots \wedge S_j(X))$$

(8) If $P,S_1,\ldots,S_j$ are relations or their negations, and m is a constant, then from

$$\begin{gathered} P(m) \\ S_1(m) \\ \vdots \\ S_j(m) \end{gathered}$$

we can deduce

$$P(X)\ I\ (S_1(X) \wedge \ldots \wedge S_j(X))$$

(9) If $P,M_1,\ldots,M_l,S_1,\ldots,S_j$ are relations or their negations, then from

$$\begin{gathered} P(X)\ I\ (M_1(X) \wedge \ldots \wedge M_l(X)) \\ S_1(X)\ A\ (M_1(X) \wedge \ldots \wedge M_l(X)) \\ \vdots \\ S_j(X)\ A\ (M_1(X) \wedge \ldots \wedge M_l(X)) \end{gathered}$$

we can deduce

$$P(X)\ I\ (S_1(X) \wedge \ldots \wedge S_j(X))$$

(10) If $P,S_1,\ldots,S_j$ are relations or their negations, and $1 \leqslant k \leqslant j$, then from

$$P(X)\ I\ (S_1(X) \wedge \ldots \wedge S_j(X))$$

we can deduce

$$P(X)\ I\ (S_1(X) \wedge \ldots \wedge S_{k-1}(X) \wedge S_{k+1}(X) \wedge \ldots \wedge S_j(X))$$

(11) If $P,S_1,\ldots,S_j$ are relations or their negations, then from

$$\begin{gathered} P(X)\ A\ (S_1(X) \wedge \ldots \wedge S_{j-1}(X)) \\ S_j(X)\ I\ (S_1(X) \wedge \ldots \wedge S_{j-1}(X)) \end{gathered}$$

we can deduce

$$P(X)\ A\ (S_1(X) \wedge \ldots \wedge S_j(X))$$

1 is identical to the deductive procedures of Gallaire *et al.*, and of logic-programming. In order to be able to make queries about deductive rules (which is not allowed by Gallaire *et al.* or by logic-programming languages, but is allowed by this database) we need an equivalent of 1 for deductive rules; this is 2. 3, 4 and 5 are the extension for explicit negation; 5 is the equivalent of 4 for deductive rules. 6 thru 11 are the extension for particular quantity; 9 is the equivalent of 8 for deductive rules; 10 and 11 deal with widening and narrowing terms, as described in section 3.1.1.

7.1.1. Examples of deductions with this system

Following are some examples of deductions with the above system, showing the compatibility between its deductive powers and those of the database. The examples are taken from section 5.

To prove "¬man(Mona_Lisa)" (example 2):

statement	*justification*
(1) work_of_art(Mona_Lisa)	stored fact
(2) ¬mortal(X) A work_of_art(X)	stored rule
(3) mortal(X) A man(X)	stored rule
(4) ¬mortal(Mona_Lisa)	by rule 1 from (1) and (2)
(5) ¬man(X) A ¬mortal(X)	by rule 3 from (3)
(6) ¬man(Mona_Lisa)	by rule 1 from (4) and (5)

To prove "¬age_0_to_2(Socrates)" (example 3):

statement	*justification*
(1) man(Socrates)	stored fact
(2) wise(Socrates)	stored fact
(3) responsible(X) A wise(X)	stored rule
(4) ¬responsible(X) A baby(X)	stored rule
(5) baby(X) A (man(X) ∧ age_0_to_2(X))	stored rule
(6) responsible(Socrates)	by rule 1 from (2) and (3)
(7) ¬baby(X) A responsible(X)	by rule 3 from (4)
(8) ¬baby(Socrates)	by rule 1 from (6) and (7)
(9) ¬age_0_to_2(Socrates)	by rule 4 from (1), (5) and (8)

To prove "¬work_of_art(X) I animal(X)" (example 7):

statement	*justification*
(1) animal(X) A man(X)	stored rule
(2) mortal(X) A man(X)	stored rule
(3) ¬mortal(X) A work_of_art(X)	stored rule
(4) animal(X) I man(X)	by rule 6 from (1)
(5) animal(X) I mortal(X)	by rule 9 from (2) and (4)
(6) ¬work_of_art(X) A mortal(X)	by rule 3 from (3)
(7) animal(X) I ¬work_of_art(X)	by rule 9 from (5) and (6)
(8) ¬work_of_art(X) I animal(X)	by rule 7 from (7)

As can be seen in these examples, the structure of the deduction is lost in the use of the above system. The classification of syllogisms into figures is not possible; and if the validity of a syllogism is determined by the possibility of reducing it to the eleven rules, rather than by the list of restrictions presented in section 3.2, then the classification of fallacies is also not possible. This means that the system captures the deductive powers of the database, but not its special capabilities.

7.2. A PARTIAL FORMAL SEMANTICS

Following is a partial formal semantics for the database. It provides a definition for consistency and semantic implication, but does not express the special capabilities of the user interaction.

This semantics, like the system in section 7.1, deals only with unary relations. Also, it deals only with types A and I.

Definition: Let F_s be a set of specific facts, and let F_g be a set of deductive rules. The "closure" of F_s with respect to F_g is a superset of F_s obtained as follows:

(1) For each rule of type I in F_g, if F_s doesn't already contain an instance of it—i.e. no

constant is known to fulfill both its subject and its predicate—then add a new constant and add specific facts asserting that it fulfills both the subject and the predicate of the rule.

(2) For each rule of type A in F_g, if no constant is known to fulfill its subject, then add a new constant and add specific facts asserting that it fulfills the subject of the rule.

(3) For each relation appearing in any rule in F_g, if no constant is known not to fulfill it, then add a new constant and add a specific fact asserting that it does not fulfill this relation.

(4) If there exists a rule of type A in F_g and a constant s such that s is known to fulfill the subject of the rule but no specific fact asserts that it fulfills the predicate, then add a new specific fact asserting that s fulfills the predicate of the rule.

(5) Repeat step 4 until no such rule and constant can be found, i.e. all direct applications of type A rules have been generated explicitly.

Step 2 is required because of the assumption that every rule is about something. Step 3 is required to prevent relations that include everything (which is not allowed by the theory of the syllogism).

DEFINITION: A set of specific facts is "consistent" if it does not contain a fact and its negation.

DEFINITION: A database is "consistent" if the closure of its specific facts with respect to its deductive rules is consistent.

DEFINITION: A set of specific facts F_s is "closed" with respect to F_g if it is equal to its own closure with respect to F_g.

DEFINITION: A set of specific facts F_s "contradicts" a fact or rule f if

(1) f is a specific fact and F_s contains its negation; or

(2) f is of type I and F_s does not contain an instance of it; or

(3) f is of type A and F_s either does not assert for any constant that it fulfills the subject of f or asserts for some constant that it fulfills the subject of f and not its predicate.

DEFINITION: A database consisting of specific facts F_s and deductive rules F_g "implies" a fact or rule f if there is no consistent superset of F_s that is closed with respect to F_g and contradicts f.

Note that the procedures in section 3, and the system in section 7.1, are limited to the equivalent of "definite deductive databases", i.e. only one relation in the predicate. This causes incompleteness with respect to the above semantics, in the treatment of explicit negation. For example, if a database contains the following facts:

$$P(X)\ A\ (S(X) \wedge M_1(X))$$
$$\neg P(X)\ A\ (S(X) \wedge M_2(X))$$
$$M_1(s)$$
$$M_2(s)$$

then, by the semantics, it implies "$\neg S(s)$"; a more general Aristotelian-logic database, that allows a disjunction of several relations in the predicate, could infer this conclusion, but the procedures in section 3, and the system in section 7.1, do not handle this case.

The procedures in section 3, and the system in section 7.1, are sound with respect to the semantics. The proof is by straightforward case analysis.

The full thesis (Mozes, 1987) contains a brief discussion of the possibility for a complete formal semantics that would also express the special capabilities of the database.

7.3. ALTERNATIVE AXIOMATISATION: REDUCTION OF THE FIGURES

An alternative to the axiomatisation in section 7.1 is the doctrine of the reduction of the figures, originated by Aristotle himself. Aristotle viewed the first figure of the syllogism as the "perfect" figure, i.e. as the most natural expression of *all* deduction. He therefore saw the four valid syllogisms of the first figure as the basic rules of deduction, to which syllogisms of other figures can be reduced. The reduction is either "direct", by conversions of the premises or the conclusion, or "indirect", by reductio ad absurdum. The reduction of the figures is presented in detail by Joseph (1916), and is more recently presented in mathematical form by Corcoran (1973).

Corcoran regards the reduction of the figures as "a natural deduction system", i.e. as a system in which deductive rules, rather than axioms, predominate. Note that the system in section 7.1 is also, by this definition, a natural deduction system.

As noted above, the user interface of the database is based on Joseph's view, which differs from Aristotle in seeing each figure as representing a different form of reasoning, and therefore regards the reduction of the figures as unnecessary. The reduction of the figures, like the system in section 7.1, captures only the deductive powers of Aristotelian logic, not the special capabilities it makes possible.

Obviously, reduction of the figure loses the classification into figures, by putting all syllogisms into the first figure. This means that the ability to structure the natural-language explanations in the most natural way is lost; for example, in example 2 in section 5, the sentence "Mona Lisa can't be a man, because then it would be mortal" will be changed into "Because Mona Lisa is immortal, it is not a man"; in the same way, in example 5, the sentence "Some animal is not a man; for example, the dogs" will be changed into "Because some animal is a dog, it is not a man". The ability to point out instances for using analogy or induction is also lost.

Again, if the validity of a syllogism is determined by the possibility of reduction, rather than by the list of restrictions, then the classification of the fallacies, and with it the ability to point out possibilities, is also lost.

The special capabilities of the database show that Aristotelian logic is more than a natural deduction system—it is a system based on the principles of actual human thought, which therefore allows making the distinctions between forms of reasoning that are necessary for good interaction with human users.

8. Implementation of the Prototype System

Following is a brief discussion of the architecture and major algorithms of the implemented prototype system.

8.1. OVERALL ARCHITECTURE

The prototype system consists of three major modules:

(1) Deduction—a module that searches the possible deduction paths, and records all paths leading to a conclusion that may interest the user.

(2) Record checking—a module that examines the records left by the deduction module, and uses them to answer the user's questions.
(3) Natural-language explanation—a module that, given a deduction path, produces a natural-language explanation of that path.

The rest of the system contains straightforward routines that interact with the user, read his commands, and parse the queries.

When accepting a new fact or rule into the database, the system always first runs the deduction module on it. If its contradictory can be deduced from the rest of the database, the user is notified and the fact or rule is rejected; if the fact or rule can itself be deduced, the user is warned and asked whether he still wants to enter it. This prevents inconsistencies, and discourages redundancies.

Since this is a prototype system, no effort was spent on sophisticated storage of the database. The database consists of two files, one containing a list of the constants and relations, the other containing a list of the facts and rules; the two files are read into memory at the start of the run, and written back again at its end.

The prototype was implemented in the C programming language.

8.2. GENERAL DESCRIPTION OF THE MAJOR ALGORITHMS

8.2.1. Deduction

The deduction module works by recursive depth-first search. Given a goal, the program finds what are the major and minor terms (respectively the predicate and subject of the goal; if the user gave a retrieval request, then a special minor term, that can match any constant, is used), and then goes over all possible major premises, i.e. all facts and rules that contain the major term. For each one, the program first decides whether it also contains the minor term (so an immediate inference can be tried); if not, it finds what is the middle term, and then looks for a fact or rule containing both the middle and minor term and tries to perform a syllogism; after that, the program decides on a sub-goal (i.e., finds what additional premise would, together with the major premise, allow deducing the goal), and goes on recursively.

All deduction paths leading to a significant conclusion (the original goal, its negation, pointing it out as a possibility, or a stronger or weaker version) are recorded. Incomplete paths are also recorded. The search ends either when the search tree is exhausted or when a strongest possible path (one leading to a sure conclusion with universal quantity, or one proving a "no" answer) is found.

In the cases handled by the database (no relations representing relationships between objects), there can be no valid deduction path using the same middle term, or the same premise, more than once; this is used to prune the search tree, by eliminating any branch using a middle term already used higher in the tree. This also prevents looping when the rules of the database contain cycles.

Premises in which the subject has more than one relation can lead to a fork in the search, when that subject becomes the middle term. The search goes on separately for each of the relations in the subject (and can fork further recursively); the record-checking module uses the results in each path of the fork to determine the total result.

Ordinarily, the search goes only in the major-to-minor direction, i.e. the program generates a subgoal for each possible major premise. This is enough for all capabilities of the database except finding missing rules; missing rules can also be found when generating subgoals for possible *minor* premises, so finding them requires doing the search in both

directions at each level of the search tree (which can make the search exponentially slower). The user has to specify in advance, before a query, that he will want to ask for missing rules, and the search is then performed in both directions.

The implementation is heuristic; it is sufficient for cases that will reasonably be needed by a human user, but is not meant to be complete with respect to Aristotelian logic. For example, from the facts:

$$M_3(X)\ I\ (M_1(X) \wedge M_2(X))$$
$$P(X)\ A\ (M_1(X) \wedge M_2(X))$$
$$S(X)\ A\ (M_1(X) \wedge M_3(X))$$

you can infer, in Aristotelian logic, or by the system in section 7.1, the conclusion "P(X) I S(X)", but the implemented prototype will not be able to infer it. A complete implementation would, if two terms overlap without any one of them containing the other (such as "$M_1(X) \wedge M_2(X)$" and "$M_1(X) \wedge M_3(X)$"), apply the deduction module recursively to try to prove a type I rule asserting the existence of objects fulfilling both terms (in this case, "$M_3(X)\ I\ (M_1(X) \wedge M_2(X))$"), and, if successful, unify the two terms by narrowing both of them; this would make it possible to handle the above case.

8.2.2. Natural-language explanations

The natural-language explanation module accepts a deduction path and, going in the minor-to-major direction (the opposite of the direction in which the path was generated), builds a sequence of syllogisms. If this is an explanation of a possibility, it is broken into two sequences of syllogisms, and the results of the two sequences are then used in one syllogism committing the fallacy of the undistributed middle.

The program keeps a set of string macros (implemented in C by calls to the "printf" function, with arguments that are calls to various functions returning strings), one for each mood and figure that either are valid or commit the fallacy of the undistributed middle; the later macros contain the word "perhaps". For each syllogism, the appropriate macro is chosen, and is then filled in by the terms and by their pronouns, thus producing a natural-language sentence explaining the syllogism.

The full thesis (Mozes, 1987) contains simplified pseudo-code listings of the deduction and natural-language explanation modules.

9. Conclusion

This paper presented a deductive database based on Aristotle's rules of the syllogism. It was demonstrated, both theoretically and by practical examples, that this database allows natural and informative interaction with the user, and that its unique capabilities in this area follow naturally from the use of Aristotelian logic.

The main conclusion we can draw from this work is that the neglect of Aristotelian logic by computer scientists is unjustified, and that more work should be done using it in applications that require good user interaction and in AI applications.

I would like to thank Professor Amir Pnueli, my thesis supervisor, for many discussions that helped me to concretise my ideas about the value of Aristotelian logic into the form of the database described in the paper. These discussions also supplied the motivation, and many important ideas, for some parts of the paper, particularly section 7. I would also like to thank Dr. Klaus-Peter Adlassnig and Dr. Franz Barachini, for generously taking the time to respond in detail to my queries about CADIAG-1 and KONSDED.

References

Adlassnig, K.-P., Kolarz, G., Scheitauer, W., Effenberger, H., Grabner, G. (1985). CADIAG: Approaches to computer-assisted medical diagnosis. *Comp. Biol. Med.* **15**, 315–335.

Atzeni, P., Parker, D. S. (1986). Set containment inference. *Proceedings of the International Conference of Database Theory*, Lecture Notes in Computer Science, Vol. 243, Springer-Verlag.

Barachini, F. (1984). *Konsistenzprüfung von Wissenbasen Medizinischer Expertensysteme*, Ph.D. dissertation, Technical University of Vienna, Vienna.

Corcoran, J. (1973). A mathematical model of Aristotle's syllogistic. *Archiv für Geschichte der Philosophie*, **5**, 191–219.

Gallaire, H., Minker, J., Eds. (1978). *Logic and Data Bases*, Plenum, New York.

Gallaire, H., Minker, J., Nicolas, J. (1984). Logic and databases: a deductive approach. *ACM Computing Surveys*, **16**, 153–185.

Joseph, H. W. B. (1916). *An Introduction to Logic*, 2nd edition. Oxford University Press, Oxford.

Lipski, W. (1979). On semantic issues connected with incomplete information databases. *ACM Transactions on Database Systems*, **4**, 262–296.

Lipski, W., Marek, W. (1977). On information storage and retrieval systems. In *Mathematical Foundations of Computer Science*, A. Mazurkiewicz & Z. Pawlak, Eds., Banach Center Publications, Vol. 2, pp. 215–259.

Lukasiewicz, J. (1951). *Aristotle's Syllogistic from the Standpoint of Modern Formal Logic*. Oxford University Press, Oxford.

Maritain, J. (1937). *An Introduction to Logic*, Sheed & Ward, London.

Mozes, E. (1987). *A Deductive Database Based on Aristotelian Logic*, CS87–18, Department of Applied Mathematics, Weizmann Institute of Science, Israel.

Ross, W. D. (Ed.) 1949. *Aristotle's Prior and Posterior Analytics*. Oxford University Press; Oxford.

Veatch, H. B. (1952). *Intentional Logic*, Yale University Press, New Haven.